Hibajelzés, hibajavítás

- A csatornán történő továbbításkor megváltozhat az üzenet.
- Ezért kibővítjük az eredeti üzenetet, hogy vételkor ellenőrizni tudjuk, történt-e elváltozás:
    - Kód+redundáns információ
- **Hibajelző** (újra-küldés)
    - Paritás bitek hozzáadása
    - Hibajavító kód megalkotása:
    - Hamming-kódok,
    - bináris konvolúciós kódok,
- Blokk kód
    - A keretek
    - *m* adatbitből, vagyis üzenetbitből (message bit) és
    - *r* redundáns bitből, vagyis ellenőrző bitből (check bit) állnak.
    - A **blokk-kódokban** (**block code**) az *r* ellenőrző bitek kiszámítása csak és kizárólag a hozzájuk tartozó *m* adatbitek függvényeként történik, pontosan úgy, mintha egy nagy táblából kikeresnénk az *m* adatbithez tartozó *r* ellenőrző bitet.
    - Ha az *m* adatbitet közvetlenül, változtatás (előzetes kódolás) nélkül küldjük el az ellenőrző bitekkel együtt, akkor **szisztematikus kódokról** (**systematic code**) beszélünk.
    - **Lineáris kódokban** az *r* ellenőrző bit az *m* adatbit lineáris függvénye, melyre gyakori példa a kizáró vagy (xor) vagy a modulo 2 összeadás. (Ez azt jelenti, hogy a kódolás folyamata mátrixszorzással vagy egyszerű logikai áramkörökkel végezhető.)
    - lineáris, szisztematikus blokk-kódok, (n,m) kód.
    - kódsebesség (code rate) vagy egyszerűen sebesség a kódszó azon része, amely a nem redundáns információt tartalmazza (tehát m/n). Zajos csatornára akár 1/2 is lehet, így a vett információ fele redundancia, míg egy jó minőségű csatornára megközelítheti az 1-et, hiszen kevés ellenőrző bitet csatolnak egy hosszú üzenethez.

MI a „hiba”?

- Az olyan helyek számát, amelyeken a két kódszóban különböző bitek állnak, a két kódszó **Hamming-távolságának** [Hamming, 1950] nevezzük.
- A jelentősége abban áll, hogy ha két kódszó Hamming-távolsága *d*, akkor *d* darab egy bitet érintő hiba/átalakítás kell ahhoz, hogy az egyik kódszó a másikba menjen át.
- Megjegyzés: természetesen nem csak bináris kódokra vonatkoztathatók a kimondottak.

Hamming távolság

- Megadva az ellenőrző bitek kiszámításának módját, meg lehet alkotni a legális kódszavak teljes listáját, és ebből a listából ki lehet keresni azt a két kódszót, melyeknek legkisebb a Hamming-távolsága. **Ez a távolság a teljes kód Hamming-távolsága.**

Hogyan észlelhetünk hibát?

- Ha m hosszúságú a bináris üzenet (információ), akkor 2^m lehetséges adatüzenet legális (ennyit tudunk megalkotni), de az ellenőrző bitek kiszámítási módja miatt nem fordul elő mind a 2^n lehetséges kódszó (mert a redundáns rész az m hosszúságú üzenet függvénye).
- r ellenőrző bit esetén csupán a lehetséges üzenetek töredéke, $2^m / 2^n$-ed vagy $1/ 2^n$-ed része lesz legális kódszó.
- Ez a különbség az, amellyel az üzenet beágyazódik a kódszavak terébe, amely lehetővé teszi, hogy a vevő kijelezze és kijavítsa az átvitel során keletkezett hibákat
- (azaz a redundáns résszel kibővített üzenetekből több van, és könnyebb felismerni, ha nem a korrekt (érvénytelen) kódszó jön át).

Hibajelző? Hibajavító?

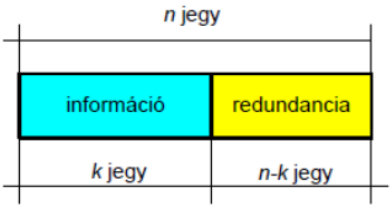
- Amikor a vevő egy érvénytelen kódszót lát, tudja, hogy átviteli hiba történt.
- Az, hogy egy blokk-kód hibajelző vagy hibajavító tulajdonságú-e, a **kód Hamming-távolságától** függ.
- Ahhoz, hogy ***t*** hibát jelezni tudjunk, ***t*+1** Hamming-távolságú kód kell, mert egy ilyen kódban ***t*** bithiba nem tudja a kódszót egy másik érvényes kódszóba vinni.
- Hasonlóan: ahhoz, hogy ***t*** hibát ki tudjunk javítani, **2*t*+1** Hamming-távolságú kód kell, mert így az érvényes kódszavak olyan távol vannak egymástól, hogy még ***t*** bit megváltozásakor is közelebb lesz az eredeti kódszó a hibáshoz, mint bármely másik érvényes kódszóhoz, így az egyértelműen meghatározható, (feltételezve, hogy nagyobb számú bithibáknak kicsi a valószínűsége.)
- Tehát legalább (azaz min) 2t+1 távolság kell legyen a két kód között, hogy biztosan el lehessen „különíteni” őket.

Példa

- d_H=5, d=5
- 5=2**t*+1=2*2+1
- t=2 bitnyi hibajavítás lehetséges.
- 0000000000
- 0000011111
- 1111100000
- 1111111111
- ha a 0000000111 kódszó érkezik, nem legális
- a vevő tudja, hogy az eredetinek 0000011111-nek kellett lennie (ez van legközelebb az érvényes kódszóhoz). (t=2)
- Azonban, ha hárombitnyi hiba a 0000000000-t 0000000111-re változtatta, akkor a kódszó nem megfelelően lesz kijavítva (t=3 lenne).

A következő jelöléseket használjuk:

- $U = (u_1, u_2, \ldots, u_M)$ - csatorna kód,
- $u_s = [u_{i1}, u_{i2}, \ldots, u_{in}]$ - i-edik vektor (kódszó) ($u_{ij} \in \{0, 1\}$)
- u_{ij} - az i-edik kódszó j-edik eleme,
- $k = \log M$ - egy kódszó információ bitjeinek száma.



- Az ilyen típusú kódokra esetenként (n, h)-kódként hivatkozunk.

Def.: Két kódszó Hamming távolságán az egymástól páronként különböző jegyek számát értjük, azaz

$$d_H(u_i, u_j) = |(k \mid u_{ik} \neq u_{jk}, 1 \leq k \leq n\}| .$$

A jelentősége abban áll, hogy ha két kódszó Hamming-távolsága ***d***, akkor ***d*** darab egy bitet érintő hiba kell ahhoz, hogy az egyik kódszó a másikba menjen át.

Def.: Egy ui kódszó Hamming-súlyán, a nemzérus komponensek számát értjük, azaz

$$W(\mathbf{u}_i) = |\{k \mid u_{ik} \neq 0,\ 1 \leq k \leq n\}| .$$

A Hamming távolságra teljesül, hogy

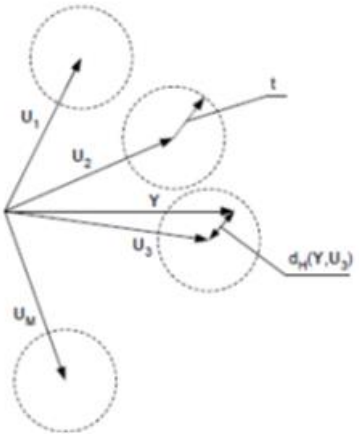
$$\begin{aligned}
&\text{(i)}\ d_H(\mathbf{u}_i, \mathbf{u}_j) \geq 0,\ d_H(\mathbf{u}_i, \mathbf{u}_j) = 0 \Leftrightarrow \mathbf{u}_i = \mathbf{u}_j, \\
&\text{(ii)}\ d_H(\mathbf{u}_i, \mathbf{u}_j) = d_H(\mathbf{u}_j, \mathbf{u}_i), \\
&\text{(iii)}\ d_H(\mathbf{u}_i, \mathbf{u}_j) \leq d_H(\mathbf{u}_i, \mathbf{u}_k) + d_H(\mathbf{u}_k, \mathbf{u}_j).
\end{aligned}$$

A Hamming távolság ezek szerint megfelel a vektortereken definiált normák feltételeinek, ami még fontos lesz, hiszen a bináris kódok tulajdonképpen vektorok .

Def.: Egy kód t hibáig korrekt, ha bármelyik kódszóban t vagy ennél kevesebb hibás szimbólumot helyesen dekódol a minimális távolság elv alapján.

Tétel: Egy kód akkor és csak akkor t korrekt t hibáig, ha d≥2t+1.

Bizonyítás: A t sugarú ui középpontú Hamming-gömböt azok a 0-1 elemű vektorok alkotják, amyelyek Hamming távolsága ui-től kisebb, vagy egyenlő, mint t.

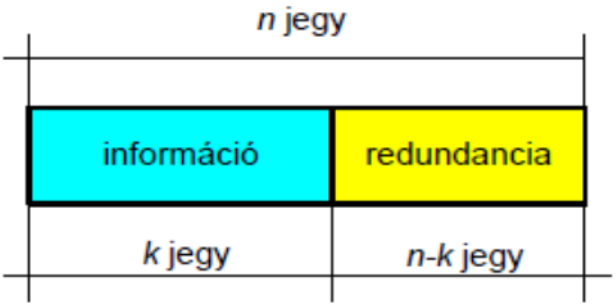


t sugarú ui középpontú Hamming gömb

Következmény: Egy adott kód hibajavító képessége t, és a kód minimális távolsága összefüggenek. Nevezetesen

$$t=[(d-1)/2]$$

Emlékeztető: $\lfloor 2{,}13 \rfloor = 2$



A blokk kódok három f"o paramétere:

M - a kódvektorok száma (kód mérete),

n - a kódszavak hossza,

d - a kódtávolság.

Az M paraméter helyett szokás az információs bitek h számát, vagy az R = h/n hányadost is használni. Szokás továbbá a blokk kódra az (n, M, d) formában is hivatkozni.

A jó hibajavító tulajdonság elérése adott M (k vagy R) és n esetén a d "*maximálását*" jelenti. Duális formában ugyanez az R hányados (k vagy M) maximalizálását jelenti adott d minimum távolság és n hosszúság esetén.

Egy adott kódszó t sugarú Hamming-gömbjében $\sum_{i=0}^{t} \binom{n}{i}$ darab vektor van. Ebből következik az alábbi

Tétel *(Hamming korlát): Tetszőleges, legfeljebb t hibát javító blokk kódra fenn kell állnia a*

$$M \sum_{i=0}^{t} \binom{n}{i} \leq 2^n \tag{3.4}$$

egyenlőtlenségnek.

Az egyenlőséget kielégítő kódokat *perfekt kódoknak* nevezzük.

A jó hibajavító tulajdonság elérése adott M (h vagy R) és n esetén a d "maxsmá1ását" jelenti. Duális formában ugyanez az R hányados (h vagy M) maximalizálását jelenti adott d minimum távolság és n hosszúság esetén.

Egy adott kódszó t sugarú Hamming-gömbjében $\sum_{i=0}^{t} \binom{n}{i}$ darab vektor van. Ebből következik az alábbi

Tétel (Hamming korlát): Tetszőleges, legfeljebb t hibát javító blokk kódra fenn kell állnia a

$$M \sum_{i=0}^{t} \binom{n}{i} \leq 2^n$$

Egyenlőtlenségnek.

Az egyenlőséget kielégítő kódokat **perfekt kód**oknak nevezzük.

A perfekt kódok kijavítanak t vagy kevesebb hibát, de egyikük sem ismer fel, vagy javít t-nél több hibát. Geometriailag ezek a kódok a Xn darab bináris vektor M darab t sugarú gömbbel történő közös részek és hézagok nélküli, un. "1egssovosabb hstö1tését" valósítják meg. A "tökéletességük" az R maximálását jelenti adott d = Xt ‡ fi és n esetén.

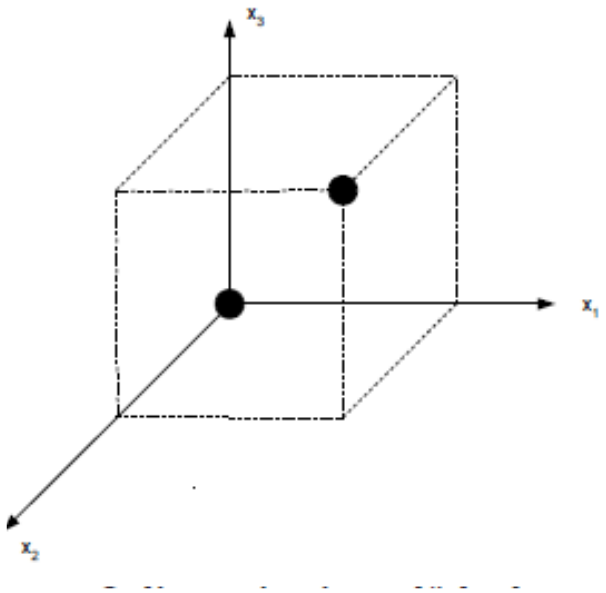
Csak három tökéletes bináris kódosztály létezik:

- páratlan hosszúságú bináris ismétlő kódok,
- Hamming kódok (fi hibát javítanak),
- Golay kód (n = X3, d = t, 3 hibát javít).

Példa: A legegyszerűbb bináris ismétlő kód hossza 3. A kód a jelet háromszor megismétli: a

’0’-hoz a ’000’, az ’1’-hez az ’111’ kódszót rendeli. A két kódszó távolsága dH ((000) , (111)) =

3. Ugyancsak igaz, hogy dmin = 3. A 3 hosszúságú bináris vektorok száma $2^3 = 8$ és ezek megfeleltethetők a 3D egységkocka csúcsainak.

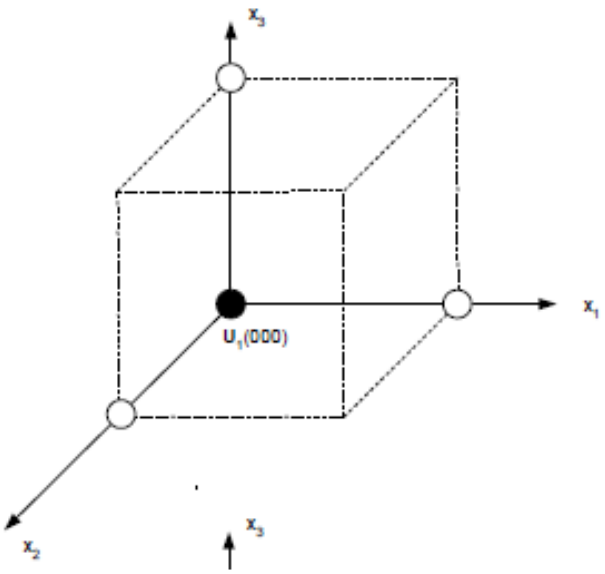


Kérdés: hogyan döntjük el, hogy a téves 001, 010, 100, 110, 101, 011 kódokat melyik jó (000 vagy 111) kódba dekódoljuk? Hova „lökjük/lökhetjük” a téves „pontokat”? Hány „lépéssel” lkjük őket a helyükre?

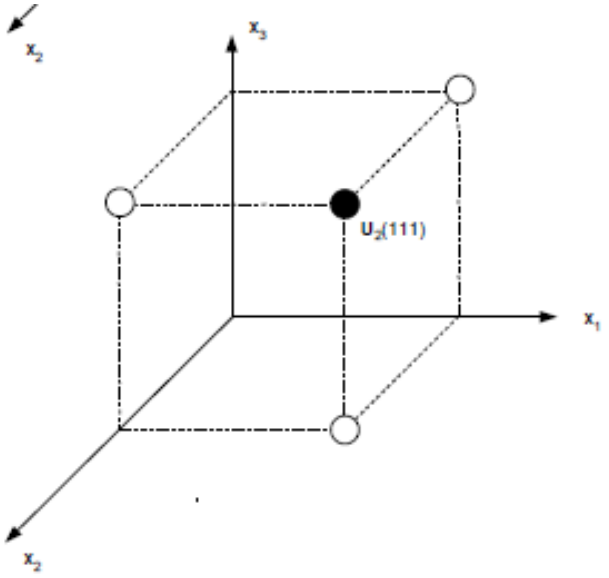
100

Kérdés: mekkora a távolsága a 000 kódtól? Mely kódok vannak még ilyen távolságra?

011



Kérdés: mekkora a távolsága az 111 kódtól? Mely kódok vannak még ilyen távolságra?



Tétel *(Plotkin korlát): Bármely blokk kódra teljesül, hogy*

$$M \leq 2^{n-2d+2}d.$$

A megfelelő blokk kód létezéséhez a Hamming és Plotkin korlátok csak szükséges, de nem elégséges feltételeket adnak. Ezért őket gyakran felső korlátoknak nevezzük. Léteznek elégséges feltételek (alsó korlátok) is egy kód létezésére adott M, n és d esetén.

Tétel (Ploktin korlát): Bármely blokk kódra teljesül, hogy

$$M \leq 2^{n-2d+2}d.$$

A megfelelő blokk kód létezéséhez a Hamming és Plotkin korlátok csak szükséges, de nem elégséges feltételeket adnak. Ezért őket gyakran felső korlátoknak nevezzük. Léteznek elégséges feltételek (alsó korlátok) is egy kód létezésére adott M, n és d esetén.

Tétel *(Gilbert korlát): Ha a paraméterek kielégítik a*

$$(M-1)\sum_{i=0}^{d-1}\binom{n}{i} < 2^n \tag{3.5}$$

egyenlőtlenséget, akkor létezik ilyen tulajdonságú blokk kód.

Ha $k = \log M$ egész szám, akkor egy sokkal erősebb alsó korlát is ismert.

Tétel *(Varshamov-Gilbert korlát): Ha a paraméterek kielégítik a*

$$M\sum_{i=0}^{d-2}\binom{n-1}{i} < 2^n \tag{3.6}$$

egyenlőtlenséget, akkor létezik ilyen tulajdonságú blokk kód.

Tétel (Gilbert korlát). Ha a paraméterek kielégitik a

$$(M-1)\sum_{i=0}^{d-1}\binom{n}{i} < 2^n$$

Egyenlőséget, akkor létezik ilyen tulajdonságú blokk kód.

Ha k=logM egész szám, akkor egy sokkal erősebb alsó korlát is ismert.

Tétel (Vaeshamov-Gilbert korlát): Ha a paraméterek kielégítik a

$$M\sum_{i=0}^{d-2}\binom{n-1}{i}<2^n$$

Egyenlőséget, akkor létezik ilyen tulajdonságú blokk kód.

Példa: Lássuk az eredeti (2 hosszúságú) kódok bővítését 5 hosszúságúra (n=5, k=2 a korábbi jelölés szerint). Túlzás ez a bővítés a 3 hosszúságú redundás résszel?

Legyen K az 1. példabeli kód, u=(0 0 0 0 0), t=1.
Az u körüli 1 sugarú gömbben a következő sorozatok vannak.

(0 0 0 0 0)
(1 0 0 0 0)
(0 1 0 0 0)
(0 0 1 0 0)
(0 0 0 1 0)
(0 0 0 0 1)

Figyeljük meg, hogy nincs a gömb elemei között a (0 0 0 0 0)-t leszámítva kódszó, ami összhangban van azzal, hogy a kód távolsága nagyobb 1-nél (3).

Ezeket a kódokat mindenképpen a 00000

kódszóként értelmezzük, ami egyértelműen a

00

eredeti üzenetet jelenti

(0 0) → (0 0 0 0 0)
(0 1) → (0 1 1 1 0)
(1 0) → (1 0 1 0 1)
(1 1) → (1 1 0 1 1)

Egy példa, ha hosszabb a kódszó (csak az elméleti korlátok figyelembevételével):

7. példa.

Állapítsuk meg, hogy van-e 5 minimális távolságú 13 hosszú perfekt bináris kód?

n=13, d=6, és $\left\lfloor \frac{d-1}{2} \right\rfloor = 2$ így a kód t=2-hibajavító.

Ha a közleményszavak k hosszúak, akkor a közleményszavak száma 2^k, ami megegyezik a kódszavak számával, $M= 2^k$.

Ha a kód perfekt, akkor a Hamming-korlát egyenlőséggel teljesül:

$$2^k\left(\binom{13}{0}+\binom{13}{1}+\binom{13}{2}\right)=2^{13} \qquad 2^k\left(1+13+\frac{12\cdot 13}{2}\right)=2^{13}$$

$$2^k \cdot 92 = 2^{13}$$

A jobb oldalon 2 hatványa áll, a bal oldalon szereplő 92 azonban nem 2 hatványa, ez az egyenlőség nem teljesülhet semmilyen k esetén. Ezért ilyen kód nem létezik

Általánosan megállapíthatjuk, hogy:

Shannon tétele alapján egy eléggé hosszú hibajavító kóddal tetszőlegesen kis hibavalószínűség érhető el, ha a kódolt bitek átviteli sebessége kisebb mint a csatorna kapacitása.

A hosszabb kóddal javul a hibajavító képessége, de nő a dekódolás bonyolultsága és ideje.

Hibakorlátozó kódolás feladatai:

- hibajelzés
- hibajavítás
  Előnyei?:
  - Forráskód - redundancia eltávolítás
  - Csatornakód - kód + redundáns rész, a rekonstruálás biztonsága érdekében
  - Zaj - a modulátor-demodulátor vonalon
  - Ismétlés elkerülése - csatornakapacitás

Hibajelzés folyamata:

- a vevő észleli a hibát
- a vevő értesíti az adót a hibáról
- (az adó általában újraküldi az üzenetet)

Hibajavítás: A vevő alkalmas bizonyos hibák javítására.

Hibrid kódolási eljárások: A vevő először hibajavítást végez, majd ellenőrzi a javítást.

Csatornakódolás kapacitástétele: (K/N) < C, ahol K: forrásszöveg egy blokkjának (részletének) hossza, N: kibővített blokk hossza, C: csatornakapacitás

Jelölések, betűk, vektorok:

k: a szimbólumok darabszáma

u = (u1, u2, …, uk): szimbólumok, üzenetvektorok, forrásABC elemei

c = (c1, c2, …, cn): üzenetvektorokból (valamilyen leképezéssel) képzett N hosszúságú kódvektorok, kódszavak

q: kódszimbólumok értékei – általában 2, mert bináris

v = (v1, v2, …, vn): vett szó, kimeneti bináris szó

Hamming távolság

- jelölés: d(c, v)
- két vektor között minimálisan megváltoztatandó elemek száma, hogy a két vektor ugyan az legyen

t = d(c, v): a hibák száma (megegyezik a Hamming távolsággal)

c': a csatorna kimenetén lévő lehetséges kódok

u': a c'-höz rendelt kódmegfelelők amik az u értékeit vehetik fel

Kódolás menete:

- adott u vektorok, K=2, q=2, q^k=4 -> 4 féle kód van (00, 01, 10, 11)

u
0 0
0 1
1 0
1 1

- kellenek c vektorok, minden kódhoz kell rendelni egy N hosszúságú kódot és ezek között a Hamming távolságnak a lehető legnagyobbnak kell lennie (mindegyiket nézve mindegyikhez), N=5

c
0 0 0 0 0
0 1 1 0 1
1 0 1 1 0
1 1 0 1 1

Dekódolás

- q^N = 32 féle kód érkezhet a csatorna kimenetére, ebből ugye csak 4 van ami eredetileg helyes, a maradék 28-hoz hozzárendeljük a hozzá legközelebb lévő kódot (Hamming távolság) a 4 közül és ez alapján dekódolunk a leggyakrabban
- $d(\mathbf{c}',\mathbf{v}) = \min_{\mathbf{c}\in C} d(\mathbf{c},\mathbf{v}).$

v	c'	u'
0 0 0 0 0	0 0 0 0 0	0 0
1 0 0 0 0	0 0 0 0 0	0 0
0 1 0 0 0	0 0 0 0 0	0 0
1 1 0 0 0	0 0 0 0 0	0 0
0 0 1 0 0	0 0 0 0 0	0 0
1 0 1 0 0	1 0 1 1 0	1 0
0 1 1 0 0	0 1 1 0 1	0 1
1 1 1 0 0	0 1 1 0 1	0 1
0 0 0 1 0	0 0 0 0 0	0 0
1 0 0 1 0	1 0 1 1 0	1 0
…	…	…
…	…	…

Hátrány hogy ehhez a folyamathoz nagyon nagy tárhely kell minden lehetséges párosítás eltárolásához.

Kódtávolság:

- minimális Hamming távolság
- $d_{min} = \min_{\mathbf{c}\neq\mathbf{c}' \quad \mathbf{c},\mathbf{c}'\in C} d(\mathbf{c},\mathbf{c}').$
- legkisebb Hamming távolság, ami a kód két legközelebbi kódszava között van

c
0 0 0 0 0
0 1 1 0 1
1 0 1 1 0
1 1 0 1 1

- , bármelyik kettőt nézzük, a legkisebb Hamming távolság 3

Hibajelzés:

- akkor tudunk hibát jelezni, ha a hiba a kódszót a kódtér nem használt vektorába viszi át.
- a biztosan jelezhető hibák legnagyobb száma kevesebb lehet, mint a két legközelebbi kódszó közötti távolság
- $t_{jel} < d_{min}$ illetve $t_{jel} = d_{min} - 1$

Hibajavítás:

- $d_{min} > 2\,t_{jav}$ illetve $t_{jav} = \mathrm{Int}\,\frac{d_{min} - 1}{2}$
- levezetés:

Hibajavítás esetén azt kérdezzük, hogy ha t a hibák száma, akkor mi biztosítja azt, hogy a **v** vett szóból a **c** küldött kódszó egyértelműen visszaállítható legyen. Ennek a formális feltétele az, hogy minden más **c'** kódszóra fennálljon a

$$d(\mathbf{v},\mathbf{c}') > d(\mathbf{v},\mathbf{c}) \qquad (7.1)$$

egyenlőtlenség, azaz egyértelműen **c** legyen legközelebb a vett szóhoz. Mivel a Hamming-távolság valóban távolság (nemnegatív, szimmetrikus és teljesíti a háromszög-egyenlőtlenséget), ezért a háromszög-egyenlőtlenség alapján

$$d(\mathbf{v},\mathbf{c}') \geq d(\mathbf{c},\mathbf{c}') - d(\mathbf{v},\mathbf{c}), \qquad (7.2)$$

így a (7.1) cél elérhető, ha a (7.2) jobb oldala nagyobb, mint $d(\mathbf{v},\mathbf{c})$, azaz

$$d(\mathbf{c},\mathbf{c}') - d(\mathbf{v},\mathbf{c}) > d(\mathbf{v},\mathbf{c}).$$

Innen a $d(\mathbf{c},\mathbf{c}') > 2d(\mathbf{v},\mathbf{c})$ átrendezett alakot kapjuk. Ez az egyenlőtlenség biztosan teljesül, ha a kódtávolságból adódó $d(\mathbf{c},\mathbf{c}') \geq d_{min}$, $\mathbf{c} \neq \mathbf{c}'$ összefüggést is figyelembe véve a

$$d_{min}/2 > d(\mathbf{v},\mathbf{c})$$

egyenlőtlenség fennáll. Összefoglalva:

7.2. tétel: Egy d kódtávolságú C kód hibajavító képessége $\mathrm{int}[(d_{min} - 1)/2]$.

Lineáris kódok:

- egyszerű matematikai leírásmód
- számos kódkonstrukció és hatékony dekódolási eljárás létezik rá
- egyszerű implementálás
- bevezetésre kerül egy G mátrix ami c=u*G számolható

$$\mathbf{G} = \begin{bmatrix} 10110 \\ 01101 \end{bmatrix}$$

u	c
0 0	0 0 0 0 0
0 1	0 1 1 0 1
1 0	1 0 1 1 0
1 1	1 1 0 1 1

- egy C bináris kódot **lineáris kódnak** nevezünk, ha C halmaz lineáris tér, azaz ha minden c, c' eleme C esetén c+c' eleme C is fennáll.
- egyszerű hibadetektálás, hibajavítás, üzenetekhez tartozó kódszavak egyszerű generálása
- $\mathbf{c} = \sum_{i=1}^{k} u_i \mathbf{g}_i$
- G: generátormátrix

- egy kódhoz számtalan generátormátrix tartozik, annyi ahány különböző bázisa lehet egy lineáris térnek
- egy kódoláshoz, üzenet kódszó összerendeléshez csak egy adott generátormátrix tartozik
- **szisztematikus kód:** ha a kódszavak első k bitje megfelel az üzenetnek
    - szisztematikus kódnál egyértelmű a generátormátrix
    - $\mathbf{G} = (\mathbf{I}_k, \mathbf{B})$,
      ahol $\mathbf{I}_k$ egy $k \times k$ méretû egységmátrix, $\mathbf{B}$ pedig egy $k \times (n - k)$ méretû mátrix.
    - **üzenetszegmens:** c első k koordinátájából álló szegmens
    - **paritásszegmens:** utolsó n-k koordinátájú szegmens
- H (n-k)xn mátrix detektálni tudja C kódszavait
    - Hc(transzponált) = 0 paritás-ellenőrző mátrix
      $\mathbf{H} = (\mathbf{A}, \mathbf{I}_{n-k})$,
    - $\mathbf{A} = -\mathbf{B}^T$,
    - $\mathbf{I}_{n-k}$ $(n - k) \times (n - k)$ méretû egységmátrix.
    - bizonyítások:
      továbbá $\mathbf{I}_{n-k}$ $(n - k) \times (n - k)$ méretû egységmátrix. A (7.6), (7.7) állítások egyszerûen igazolhatók: (7.3) és (7.5)-böl a tetszõleges összetartozó $\mathbf{c}, \mathbf{u}$ párra a

      $$\mathbf{Hc}^T = \mathbf{H}(\mathbf{uG})^T = \mathbf{HG}^T\mathbf{u}^T = \mathbf{0}$$

      egyenlõségláncot kapjuk, ahonnan

      $$\mathbf{HG}^T = \mathbf{0} \qquad (7.8)$$

      összefüggés adódik. A (7.8) egyenlõségbe behelyettesítve a (7.4) valamint a (7.6) összefüggéseket:

      $$\mathbf{HG}^T = (\mathbf{A}, \mathbf{I}_{n-k})(\mathbf{I}_k, \mathbf{B})^T = \mathbf{A} + \mathbf{B}^T = \mathbf{0}$$

      alapján (7.7) igazolására jutottunk.
- e = c - v: hibavektor
- $\mathbf{Hv}^T = \mathbf{H}(\mathbf{c}+\mathbf{e})^T = \mathbf{Hc}^T + \mathbf{He}^T = \mathbf{He}^T$,
- s = e*H(transzponált): hibavektor szindróma
- **szindrómadekódolás:**
  Ezek után a *szindrómadekódolás* lépései a következõk:
  1. A **v** vett szónak megfelelõ s szindróma kiszámítása.
  2. A dekódolási táblázat **s**-nek megfelelõ sorából a becsült **e** hibavektor kiolvasása.
  3. A **c'** = **v**-**e** dekódolási lépés elvégzése.
  4. Az **u'** dekódolt üzenet **c'**-hez rendelése.

Ismétléses kód

- az üzenet k=1 méretű, ezt ismételjük n-szer
- a kódtávolság n
- hibajavító képesség (n-1)/2

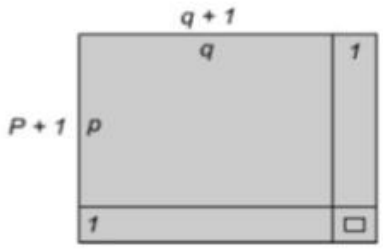
Paritáskód

- kódtávolság: 2
- 1 hiba detektálására alkalmas
- egy u üzenethez az üzenet bitjeinek megfelelő páros vagy páratlan paritásbitet illesztve kapjuk a kódszót.

Kétdimenziós paritáskód

Rendezzük a k üzenetbitet egy **U** $p \times q$ méretű mátrixba, azaz $k = pq$. Képezzünk soronként, ill. oszloponként paritást, és az így adódó paritásbiteket írjuk a sor következő elemeként a sor végére, ill. az oszlop alá (7.3. ábra). Ekkor egy **C** $(p+1)\times(q+1)$ méretű mátrixot kaphatunk, ha a mátrix még nem definiált jobb alsó sarokelemét is megadjuk. Legyen ez a sarokelem a "paritások paritása", azaz a **C** mátrix utolsó sorában vagy utolsó oszlopában álló paritásbitek paritása (könnyen látható, hogy mindegyik úton azonos érték kerül a jobb alsó sarokba).

A kódtávolság 4, amit a következőképp láthatunk be. Tekintsük a páros paritású esetet. Tartalmazzon az **U** üzenetmátrix egyetlen 1-et. Ekkor az 1-nek megfelelő sorban, ill. oszlopban a paritásbit 1, továbbá a jobb alsó sarokba kerülő paritásbit is 1. Tehát ezen kódszóban 4 db. 1 lesz, azaz a csupa zérus kódszótól való távolsága 4 (a Hamming-távolságot a mátrixok megfelelő koordinátáinak egybevetésével nyerjük). Tetszőleges kód esetén két tetszőleges kódszó Hamming-távolsága nyilván azonos a különbségükben található nemzérus elemek számával. Továbbá lineáris kódok esetén a kódszavak különbsége is kódszó. Következésképpen a kódszópárok közötti minimális távolság azonos a legkevesebb 1-et tartalmazó nemzérus kódszó 1-eseinek a számával. Az esetek végigpróbálásával könnyen belátható, hogy a kétdimenziós paritáskód esetén 4-nél kevesebb 1-et tartalmazó nemzérus kódszó nincsen, így a a kódtávolság valóban 4.



7.3. ábra. Kétdimenziós paritáskód

Hamming kód

A szindrómaszámítás $\mathbf{s} = \mathbf{eH}^T$ definíciójára tekintve láthatjuk (7.2. ábra), hogy azon **e** vektorhoz, amely (000...1...0) alakú, azaz egyetlen 1-et tartalmaz az i-edik koordinátáján, a **H** mátrix i-edik oszlopa rendelődik szindrómaként. Következésképpen, ha azt szeretnénk, hogy az n különböző ilyen alakú hibavektorhoz különböző szindrómák tartozzanak, a **H** oszlopait egymástól és nullától különbözőknek kell választani. Ez a választás tehát garantálja, hogy a **H** mátrixnak megfelelő kód minden 1 hibát tartalmazó meghibásodást javítani tudjon. A **H** mátrixnak megfelelő kódot, ill. **G** generátormátrixot úgy kaphatjuk meg egyszerűen, hogy a **H** mátrixot szisztematikus (7.6) alakban építjük fel, majd a (7.7) összefüggést használjuk.

3.3. Lineáris kódok

Legyen $V_2^{(n)} = \{\mathbf{x} = [x_1, x_2, \ldots, x_n] \mid x_i \in GF(2)\}$. Válasszunk ki $k < n$ darab lineárisan független $\mathbf{g}_1, \mathbf{g}_2, \ldots, \mathbf{g}_k \in V_2^{(n)}$ vektort! Ez mindig lehetséges, mert $V_2^{(n)}$ egy n-dimenziós vektortér. Legyen

$$U = \left\{ \sum_{i=1}^{k} a_i \mathbf{g}_i \mid a_i \in GF(2) \right\}. \tag{3.16}$$

Könnyen látható, hogy U elemeinek száma 2^k és U a $V_2^{(n)}$ vektortér k-dimenziós altere.

Az U-t azonosítjuk a lineáris kód fogalmával. Eszerint az (n, k) típusú bináris lineáris kód az n hosszúságú vektorok terének tetszőleges k-dimenziós altere. A kódszavak száma ekkor $M = 2^k$.

Definiáljuk a G $k \times n$ típusú mátrixot a következőképpen:

$$\mathbf{G} = \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix} = \begin{bmatrix} g_{11} & g_{12} & \cdots & g_{1n} \\ g_{21} & g_{22} & \cdots & g_{2n} \\ \vdots & \vdots & & \vdots \\ g_{k1} & g_{k2} & \cdots & g_{kn} \end{bmatrix}.$$

Vezessük be a k komponensű $\mathbf{a} = [a_1, a_2, \ldots, a_k]$ információs vektort! Ekkor az U lineáris kód tetszőleges $\mathbf{u}$ kódszava felírható a következő formában:

$$\mathbf{u} = \sum_{i=1}^{k} a_i \mathbf{g}_i = [a_1, a_2, \ldots, a_k] \begin{bmatrix} \mathbf{g}_1 \\ \mathbf{g}_2 \\ \vdots \\ \mathbf{g}_k \end{bmatrix} = \mathbf{aG}.$$

A $\mathbf{G}$ mátrixot a lineáris kód generátor mátrixának nevezzük.

A lineáris kódok két fontos tulajdonsága:

(i) Minden lineáris kód tartalmazza a $\mathbf{0} = [0, 0, \ldots, 0]$ zérus kódszót.

(ii) Egy lineáris kód kódszavainak összege ugyanazon kód kódszava lesz:

$$\mathbf{u}_1, \mathbf{u}_2 \in U \Rightarrow \mathbf{u}_1 + \mathbf{u}_2 \in U. \tag{3.17}$$

Tétel: *Lineáris kódok minimális távolsága megegyezik a nemzérus kódszavak súlyainak minimumával, azaz*

$$d = \min_{\mathbf{u}_i \neq \mathbf{0}} W(\mathbf{u}_i). \tag{3.18}$$

Ez azt jelenti, hogy lineáris kódok esetén a minimum távolságot $M - 1$ nemzérus szó ellenőrzésével kaphatjuk meg, ahelyett, hogy a Hamming-távolságot kiszámolnánk $M(M-1)/2$ kódszó párra.

Tetszőleges lineáris kódot áttranszformálhatunk egy ekvivalens kódba (ugyanolyan M, n, d értékekkel), amelynek generátormátrixa

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & \dots & 0 & g_{1,k+1} & \dots & g_{1,n} \\ 0 & 1 & & 0 & g_{2,k+1} & \dots & g_{2,n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & & \vdots \\ 0 & 0 & \dots & 1 & g_{k,k+1} & \dots & g_{k,n} \end{bmatrix} = [\mathbf{I}_k|\mathbf{P}] \tag{3.19}$$

alakú, ahol $\mathbf{I}_k$ $k \times k$ típusú egységmátrix, $\mathbf{P}$ pedig $k \times (n-k)$ típusú mátrix. Ebben az esetben un. *szisztematikus* kódról beszélünk, mert a kódszó információ tartalma mindig a kódszó első k szimbólumában van:

$$\mathbf{u} = \mathbf{aG} = \mathbf{a}\,[\mathbf{I}_k|\mathbf{P}] = [\mathbf{a}|\mathbf{aP}]\,. \tag{3.20}$$

Tekintsünk most egy szisztematikus U lineáris kódot a $\mathbf{G} = [\mathbf{I}_k|\mathbf{P}]$ $k \times n$ tipusú generátormátrixával. A $(n-k) \times n$ típusú

$$\mathbf{H} = \left[-\mathbf{P}^T|\mathbf{I}_{n-k}\right] \tag{3.21}$$

mátrixot a kód *paritásellenőrző mátrixának* nevezzük.
Tétel: $\mathbf{u} \in U$ akkor és csak akkor, ha $\mathbf{uH}^T = \mathbf{0}$.
Bizonyítás:

$$\mathbf{uH}^T = \mathbf{aGH}^T = [\mathbf{a}|\mathbf{aP}] \begin{bmatrix} -\mathbf{P} \\ \mathbf{I}_{n-k} \end{bmatrix} = -\mathbf{aP} + \mathbf{aP} = \mathbf{0}. \tag{3.22}$$

Az állítás azt jelenti, hogy a lineáris kód a H mátrix nulltere, illetve azt, hogy egy mátrix-vektor szorzással ellenőrizhetjük a kódszó valódiságát.
Megjegyzés: *A $V_2^{(n)}$ vektortér esetén*

$$\mathbf{H} = \left[-\mathbf{P}^T|\mathbf{I}_{n-k}\right] = \left[\mathbf{P}^T|\mathbf{I}_{n-k}\right]. \tag{3.23}$$

Tétel: *Az U lineáris kód minimumtávolsága akkor és csak akkor d, ha a H paritásellenőrző mátrix bármely $d-1$ oszlopa lineárisan független, de van d oszlopa, amely már lineárisan összefüggő (H oszloprangja $d-1$).*

3.3.1. Hamming kódok

Vegyük az összes nemzérus m jegyű bináris (pozitív egész) számot és írjuk be őket egy mátrixba oszlopként valamilyen (mondjuk növekvő) sorrendben. A kapott $m \times (2^m - 1)$ méretű mátrix lesz az $n = 2^m - 1$ hosszúságú Hamming-kód $\mathbf{H}$ paritásellenőrző mátrixa. Ezt az oszlopok cseréjével átrendezzük a $\mathbf{H} = \left[-\mathbf{P}^T|\mathbf{I}_{n-k}\right]$ alakba, ahonnan a Hamming-kód generátormátrixa $\mathbf{G} = [\mathbf{I}_k|\mathbf{P}]$ már könnyen megkapható. Az $n-k = m$ egyenlőségből kapjuk, hogy az információs szimbólumok száma $k = n - m = 2^m - m - 1$, vagyis a Hamming-kód egy $(2^m - 1, 2^m - m - 1)$ tipusú lineáris kód.

Példa: a bináris $(7,4)$ Hamming-kód esetén az $1-7$ számok növekvően sorbarendezve adják a

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 0 & 1 & 0 & 1 \\ 0 & 1 & 1 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 1 & 1 & 1 & 1 \end{bmatrix}$$

paritásellenőrző mátrixot. Ha szisztematikus Hamming-kódot akarunk, akkor átrendezzük az oszlopokat úgy, hogy az $m \times m$-es egységmátrix a jobboldalon legyen:

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 1 & 1 & 1 & 0 & 0 \\ 1 & 1 & 1 & 0 & 0 & 1 & 0 \\ 1 & 1 & 0 & 1 & 0 & 0 & 1 \end{bmatrix}.$$

Ebből a $GF\,(2)$ tulajdonságainak ((3.15)) figyelembevételével kapjuk, hogy a generátormátrix

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 0 & 0 & 1 & 1 & 1 \\ 0 & 1 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 1 & 0 & 1 & 1 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 1 \end{bmatrix}.$$

A Hamming-kódok $\mathbf{H}$ paritásellenőrző mátrixának bármely két oszlopa különböző. Ezért bármely két oszlop lineárisan független (csak nemzérus oszlopok vannak). Bármelyik oszlop azonban lineáris kombinációja másik kettőnek. Következésképpen a Hamming-kódok minimális távolsága 3 és hibajavító képessége 1 hiba. A Hamming kódok perfekt kódok.

Tétel: *A Hamming-kódok perfekt egy hibát javító kódok.*

Léteznek nem bináris Hamming-kódok is. Néhány bináris Hamming-kód paramétere:

n	7	15	31	63	127
k	4	11	26	57	120
$n-k$	3	4	5	6	7
$R = k/n$	4/7	11/15	26/31	57/63	120/127

.

3.3.2. Kiterjesztett és röviditett kódok

Tetszőleges (n,k) típusú lineáris U kód kiterjeszthető $(n+1,k)$ típusú lineáris kóddá egy un. *paritásellenőrző jegy* hozzáadásával. Legyen $\mathbf{u} = (u_1, u_2, \ldots, u_n) \in U$. A kiterjesztett U' kód megfelelő kódszava ekkor

$$\mathbf{u} = (u_1, u_2, \ldots, u_n, u_{n+1}), \tag{3.24}$$

ahol

$$u_{n+1} = u_1 + u_2 + \cdots + u_n \pmod 2. \tag{3.25}$$

Emiatt

$$u_1 + u_2 + \cdots + u_n + u_{n+1} = 0 \pmod 2 \tag{3.26}$$

3.3.4. Lineáris kódok szindróma dekódolása

A lineáris kódok népszerűségének egyik oka az, hogy a minimális Hamming-távolságon alapuló dekódolás erőforrás kímélő módon implementálható számos esetben.

A dekódolás direkt formája (az összes kódszó tárolása a memóriában és mindegyik összehasonlítása a megfigyelt $\mathbf{Y}$ vektorral) gyakran nem elfogadható a szükséges memória méret, vagy számítási idő okán.

Példa: A GPS NAVSTAR $(32, 26)$ paraméterű Hamming-kód dekódolásának direkt implementációja 256 Mbyte memóriát igényelne. 100 [ns/egy memória elérés] sebességű processzor esetén a dekódolás 6 másodpercet igényelne, mialatt több mint egy kódszó kerül leadásra másodpercenként.

Vezessük be az $(n-k)$ elemű

$$\mathbf{s} = \mathbf{y}\mathbf{H}^T \tag{3.27}$$

vektort, ahol $\mathbf{H}$ az U kód paritásellenőrző mátrixa. Az $\mathbf{s}$ vektort *szindrómának* (vagy *szindróma vektornak*) nevezzük. Összesen 2^{n-k} különböző szindrómavektor lehet egy bináris kódban.

Írjuk fel az eredeti $\mathbf{u}$ kódszó $\mathbf{y}$ megfigyelt értékét az

$$\mathbf{y} = \mathbf{u} + \mathbf{e} \tag{3.28}$$

alakban, ahol $\mathbf{e}$ a *hibavektor*. A

$$d_H(\mathbf{y}, \mathbf{u}) = W(\mathbf{y} - \mathbf{u}) = W(\mathbf{e}) \tag{3.29}$$

tulajdonság miatt a minimum távolságon alapuló dekódolást másképpen is megfogalmazhatjuk:

Az $\mathbf{y}$ dekódolásához egy olyan minimális súlyú e hibavektort kell találnunk, amelyet az $\mathbf{y}$ vektorból kivonva egy kódszót kapunk:

$$W(\mathbf{e}) \to \min,$$
$$\mathbf{y} - \mathbf{e} \in U.$$

Az $\mathbf{u}\mathbf{H}^T = \mathbf{0}$ $(\mathbf{u} \in U)$ tulajdonság miatt $\mathbf{y}H^T = (\mathbf{u} + \mathbf{e})H^T = \mathbf{e}H^T$. Tehát a szindróma értékét a hibavektor határozza meg. Bármely két hibavektor, amelynek különbsége egy kódvektor, ugyanazt a szindrómát adja. Ezért pontosan 2^k hibavektor adja ugyanazt a szindrómát. Ezeknek a vektoroknak a halmazát *cosetnek* (mellékosztálynak) nevezzük. A cosetek száma megegyezik a szindrómák 2^{n-k} számával.

A szindróma számítás meghatározza a hibavektor cosetjét, vagyis 2^{-k}-szorosára csökkenti a hibavektorok összehasonlításának a számát. Ha meghatároztuk a cosetet, akkor már csak a minimális súlyú vektort kell benne megkeresni. Az ilyen vektort a coset *leader*-jének nevezzük. A coset leader nem egyértelmű. Az összes (2^{n-k} darab) coset leadert előre ki lehet számolni és egy un. "look-up" táblázatban el lehet helyezni. Ez jelentős nyereség a számítási erőforrások tekintetében.

Az előző (GPS NAVSTAR) példa esetében 2^{-21}-szeresére csökkenne az erőforrásigény. Bináris Hamming-kód esetén azonban ez nem szükséges, mert a számított szindróma azonnal megadja a torzított jelek pozícióját (bináris formában).

A szindróma dekódolás lépései:

1. Az $\mathbf{y}$ megfigyelt vektor esetén számítsuk ki az $\mathbf{s} = \mathbf{y}\mathbf{H}^T$ szindróma vektort!
2. Adott $\mathbf{s}$-hez keressük meg az $\mathbf{e}$ hibavektort a coset leaderek között!
3. Az $\mathbf{e}$ vektort az $\mathbf{y}$-ból kivonva kapjuk a dekódolt kódszót.

Tétel: *A fenti szindróma dekódolás minimális távolság típusú.*

Példa: Tekintsünk egy K bináris kódot, amelynek generátor mátrixa ($k = 2$, $n = 4$)

$$\mathbf{G} = \begin{bmatrix} 1 & 0 & 1 & 0 \\ 0 & 1 & 1 & 1 \end{bmatrix} = [\mathbf{I} \mid \mathbf{P}] .$$

A megfelelő paritás ellenőrző mátrix ($\left[-\mathbf{P}^T \mid \mathbf{I}\right] = \left[\mathbf{P}^T \mid \mathbf{I}\right]$ miatt):

$$\mathbf{H} = \begin{bmatrix} 1 & 1 & 1 & 0 \\ 0 & 1 & 0 & 1 \end{bmatrix} .$$

Legyenek a kódszavak: 0000, 1010, 0111, 1101. Vizsgáljuk az $\mathbf{y}\mathbf{H}^T$ értékeket az összes 4 bit hosszúságú jelsorozatra (a műveletek mod 2 értendők!):

$$[0,0,0,0]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [0,0]\,, \quad [0,0,0,1]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [0,1]\,,$$

$$[0,0,1,0]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [1,0]\,, \quad [0,0,1,1]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [1,1]\,,$$

$$[0,1,0,0]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [1,1]\,, \quad [0,1,0,1]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [1,0]\,,$$

$$[0,1,1,0]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [0,1]\,, \quad [0,1,1,1]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [0,0]\,,$$

$$[1,0,0,0]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [1,0]\,, \quad [1,0,0,1]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [1,1]\,,$$

$$[1,0,1,0]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [0,0]\,, \quad [1,0,1,1]\begin{bmatrix} 1 & 0 \\ 1 & 1 \\ 1 & 0 \\ 0 & 1 \end{bmatrix} = [0,1]\,,$$

$$[1,1,0,0]\begin{bmatrix}1&0\\1&1\\1&0\\0&1\end{bmatrix}=[0,1]\,,\quad [1,1,0,1]\begin{bmatrix}1&0\\1&1\\1&0\\0&1\end{bmatrix}=[0,0]\,,$$

$$[1,1,1,0]\begin{bmatrix}1&0\\1&1\\1&0\\0&1\end{bmatrix}=[1,1]\,,\quad [1,1,1,1]\begin{bmatrix}1&0\\1&1\\1&0\\0&1\end{bmatrix}=[1,0]\,.$$

Mint látható 4 szindróma vektor van, amelyekhez tartozó 4 bites "vektorok", azaz coset, a következők:

szindróma	00	01	10	11
coset	0000	0001	0010	0011
	0111	0110	0101	0100
	1010	1011	1000	1001
	1101	1100	1111	1110

A kiválasztott coset leadereket (minimális súlyú vektorokat) aláhúzás jelöli. Vegyük észre, hogy az 10 szindrómához tartozó cosetben két minimális súlyú vektor is van! Tegyük fel, hogy vesszük az $\mathbf{y}=1111$ kódszót. A hozzá tartozó szindróma $\mathbf{s}=10$. A megfelelő coset leader $\mathbf{e}=0010$ és $\mathbf{y}$ dekódolt értéke: $\mathbf{y}-\mathbf{e}=1101$. Ha coset leaderként az $\mathbf{e}=1000$ értéket vettük volna, akkor a dekódolt érték 0111 lenne.

3.3.5. Polinomok $GF(2)$ felett

Véges testek felett is értelmezhetünk (formális) polinomokat. A $GF(2)$ test felett értelmezett (formális) polinomot az

$$a(z)=a_{n-1}z^{n-1}+a_{n-2}z^{n-2}+\cdots+a_0=\sum_{i=0}^{n-1}a_iz^i$$

előírás definiálja, ahol az $a_0,a_1,\ldots,a_{n-1}$ együtthatók $GF(2)$-beli elemek, a z pedig egy formális változó. A z-nek (egyelőre) nem adunk $GF(2)$-beli értéket. Definíció szerint $z^0=1$.

A $GF(2)$ feletti formális polinomok műveletei:

$$a(z)+b(z)=\sum_i(a_i+b_i)z^i,\quad \alpha a(z)=\sum_i\alpha a_iz^i\quad(\alpha\in GF(2))\,,$$

ahol az együttható műveletek $GF(2)$-ben, azaz mod 2 értendők. Eszerint

$$a_i+b_i=c_i\mod 2,\quad \alpha a_i=d_i\mod 2.$$

Példa: $a(z)=z^4+z^2+z$, $b(z)=z^6+z^3+z^2+1$,

$$a(z)+b(z)=z^6+z^4+z^3+2z^2+z+1=z^6+z^4+z^3+z+1.$$

Információ és kódelmélet 2.zh

Tömörítés

Veszteségmentes tömörítés

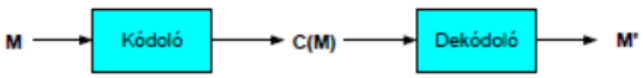
- Futáshossz kódolás
  - PackBits
  - RLE (pl a PCX használja)
- Minimális redundaciájú kódolás
  - Huffmann (figyelembe veszi a gyakoriságokat)
  - Aritmetikai kódolás
- Lexikai kódolás: DEFLATE, LZ77, LZ78, LZV, más LZ tömörítési eljárások
- Burrows-Wheeler-transzformáció (blokkrendezési feldolgozás, amely a tömörítést egyszerűbbé teszi)

Veszteséges tömörítés

- Diszkrét cosinus-átalakításokon alapuló kódolások (MP3, MPEG, JPEG)
- Fraktáltömörítés (Fraktálátalakítás)
- Hullámtömörítés

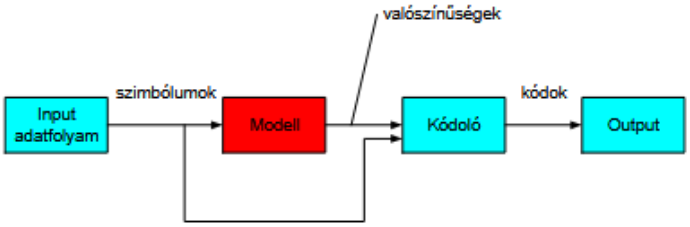
Alapfogalmak (veszteségmentes tömörítés)

- **Adattömörítés:** egy fájl méretének csökkentése, hogy a kevesebbet tárhelyet foglaljon vagy az adatátviteli idő csökkenjen.
  - Megvalósítható, mert a legtöbb fájl redundanciákat tartalmaz
- **Üzenet (M):** a bináris adatfájl, amit tömöríteni akarunk
- **Kódoló:** az üzenet (M) egy tömörített reprezentációját (C(M)) állítja elő, ami lehetőleg kevesebb bitet használ
- **Dekódoló:** visszaadja M üzenetet, vagy annak egy M' közelítését

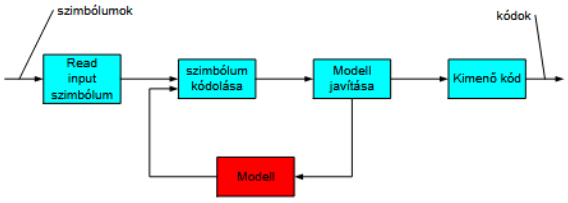


- **Kompresszió hányados:** C(M) tömörített üzenet bitjeinek száma / M üzenet bitjeinek száma
- **Veszteségmentes tömörítés:** A dekódoló program pontosan előállítja az eredeti M üzenetet.
  - Tipikus kompresszió hányados: 50-75% vagy kevesebb.
  - Lehet veszteségmentesen tömöríteni az összes fájlt!
  - Fontosabbak: futamhossz tömörítés (RLE), PPM (prediction by partial metching)
- **Veszteséges tömörítés:** A dekódoló program nem képes pontosan előállítani az eredeti M üzenetet

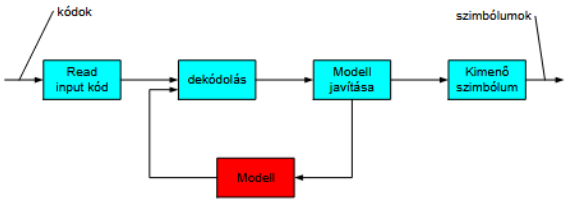
- **Tömörítő algoritmus**ok két fontos komponense:
  - Modell: az üzenetek valószínűségi eloszlását, vagy valamilyen szerkezeti összefüggését adja meg
  - Kódoló: a modell ismereteit próbálja kihasználni. Általában általános kódolók, pl Huffman kód vagy aritmetikai kód.
  - Adattömörítés = modell + kódoló
  - A modell generálja a lehetséges üzenetek valószínűségeit és a kódoló ezeket használja a kódszó előállítására. A modellnek mindkét oldalon ugyanannak kell lennie. Statisztikai modell ábrája:



- **Statisztikai modellek osztályozása:**
  - Statikus modell: Ugyanaz a modell minden szövegre. Gyors. Nem optimális, mert a különböző szövegek különböző statisztikai tulajdonságokkal rendelkeznek. pl: ASCII, Morse kód
  - Dinamikus modell: A szövegen alapuló modellt generál, előfutás szükséges a modell generálásához. A modellt is közvetíteni kell. pl: Huffman
  - Adaptív modell: Progresszíven tanulja és módosítja a modellt a szöveg olvasásával párhuzamosan. A pontosabb modellezés jobb tömörítést eredményez. A dekódolás a kezdéskor el is kell hogy induljon. pl: LZW



általános adaptív tömörítés



általános adaptív kitömörítés

Futamhossz tömörítés (RLE = run-length encoding)

- Egyszerű technika
- Az üzenetben található, hosszasan ismétlődő karaktereket egyetlen értékként/számként tárolják, az eredeti teljes karaktersorozat helyett.

- Ez leginkább sok hosszú karaktersorozattal rendelkező adat esetén hasznos. pl: egyszerű grafikus képek, mint ikonok, vonalrajzok és animációk. Nem hasznos azonban olyan adatokra, amelyeknél nincs sok ilyen karaktersorozat, hiszen jelentősen megnövelheti a fájlméretet.
- Akár a fájl növekedése is lehetséges, ha a futamok rövidek.
- Legfontosabb alkalmazásai:
  - JPEG
  - ITU-T T4 Group 3 fax: olyan legfeljebb 1728 pixel széles képeket továbbít, amelyek csak fekete és fehér pixelekből állnak. Huffman kódokat használ.

LZV tömörítés

- Szótár típusú kódoló (az adatrészek közötti ismétlődési minták megfigyelésén alapul)

Szótár típusú kódolók változatai:

- Statikus szótár: Gyakran előforduló frázisokat, digramokat (kétbetűs kombinációkat), vagy n-gramokat tartalmaz fixen. Nagy mérete lehet és kicsi kompressziója.
- Félig adaptív szótár: A kódoló létrehoz egy szótárat, a tömörítendő üzenethez illesztve. Ezt együtt tároljuk és továbbítjuk az üzenettel.
  - Két fázis: 1: szótár felépítése, 2: adattömörítés
  - Probléma az "optimális szótár" felépítése, amely NP-teljes feladat. Közel optimális algoritmusok azonban léteznek.
- Adaptív szótár: A kódolás és a szótár felépítése egyszerre történik. Nem szükséges a szótárat tárolni vagy továbbítani, mert a dekódoló is fel tudja építeni.

LZ77

- A tömörítendő szöveget egy fix méretű csúszóablakon keresztül dolgozza fel. A kódolás a csúszóablakban történik, az ablakon kívülre nem hivatkozik.
- A csúszóablak jobbra mozog, törli a már kódolt szöveget és hozzávesz egy új, még kódolatlan szövegrészt.
- Csúszóablak részei:
  - A már kódolt search buffer
  - A még kódolatlan lookahead buffer
- Az eljárás a search bufferben keresi a lookahead buffer elejének (prefixének) lehető leghosszabb illeszkedő darabját és szótárba teszi ennek kezdő pozícióját, a karaktersorozat hosszát és az első nem illeszkedő karaktert.
- Példán keresztül: (126.o) http://siva.bgk.uni-obuda.hu/~laufer/bevinfo_tankonyv/Informatika%20alapjai%20jegyzet.pdf
- Bizonyos feltételek mellett az LZ77 algoritmus aszimptotikusan optimális.
- Továbbfejlesztései
  - LZR: A mutató bárhová mutathat a kódolt területen, a csúszóablak nem korlátozza. A pozíciót és a hosszúságot változó hosszúságú kóddal reprezentálják.

- LZSS: Az LZ77-ben az n hosszat az első nem illeszkedő karakter definiálja. Ez a változat ezt eltörli és fix hosszúságú kódszavakat keres.
- LZB: A referencia és a hossz kódolást jóval kifinomultabban csinálja.
- LZH: Huffman kódot használ a mutató tömörítésére.

Tömörítés hatékonysága (veszteségmentes esetén)

- A tömörítés hatékonysága a kiinduló fájl méretének és a tömörített fájl méretének figyelembevételével határozható meg.
- Kompressziós együttható: A tömörítés kimenetének és bemenetének aránya
  - Méret a tömörítés után / Méret a tömörítés előtt
- Kompressziós faktor:
  - Méret a tömörítés előtt / Méret a tömörítés után
- Megtakarítási százalék: A zsugorodást mutatja százalékban
  - (Méret a tömörítés előtt / Méret a tömörítés után) * Méret a tömörítés előtt (%)

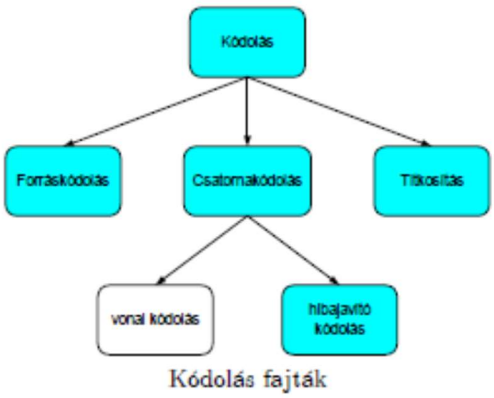
Titkosítás

I. ISMÉTLÉS

Kommunikációs csatorna

Egyenlő időközönként egymásután következő jelek (csatorna kódjelek) vihetők át rajta

Eredeti üzenet kódolása – forráskódolás (Információ és entrópia mérőszámok)



Kódolás fajták

- Relatív gyakoriság
- Átlagos kódhossz
- Egy betűre jutó költség

Csatornakódolás (kölcsönös entrópia)

A csatornakódolás problémái:

- Zaj, azaz a csatorna be- és kimenete különbözik (Hamming távolság,
- hibafelismerő és hibajavító kódok)
- Továbbítás/tárolás költségeinek csökkentése (tömörítési eljárások)
- Titkosítás (adatvédelem az illetéktelen „kódolvasók” aktív vagy passzív
- támadásaitól a továbbításkor)

Miért fontos a titkosítás?

Az egymástól távol lévő emberek közötti bizalmas kommunikáció igénye vezetett a titkosítás kialakulásához. Egy titkosítás, akkor nevezhető biztonságosabbnak, ha minél nagyobb a titkosítás feltöréséhez szükséges idő.

Alapfogalmak:

- Az üzenet küldője az a személy, aki valakivel meg akar osztani valamit.
- A titkosítás előtti üzenetet nyílt üzenetnek nevezzük.
- Az üzenet küldője a nyílt üzenetet titkosítja a kulcs segítségével, aminek eredményeként előáll a titkos üzenet.
- A kulcs egy olyan adat, ami nélkül sem a titkosítás, sem a visszafejtés nem lehetséges hagyományos módszerekkel. Normális esetben tehát a titkos üzenet értelmezhetetlen és visszafejthetetlen a kulcs birtoklása nélkül.
- A titkosító algoritmusnak az a feladata, hogy előállítsa a titkos üzenetet.
- Az üzenet fogadója az a személy, akinek a nyílt üzenetet el szeretnénk juttatni titkosított formában. Az üzenet küldőjétől megérkezett titkos üzenetből a kulcs birtokában visszaállítható a nyílt üzenet. Ezt a folyamatot nevezzük visszafejtésnek.

Milyen típusú támadások léteznek?

A támadás lehet:

- Passzív: A passzív módszer az un. lehallgatás, amikor a támadó a nyílt csatornán átmenő üzenetek birtokába kerül. Ha ezek birtokában a kulcsot meg tudja határozni, akkor azt mondjuk, hogy sikerült feltörni a rejtjelez½o algoritmust. Ezután lehetővé válik a további szövegek megfejtése is.
- Aktív: Az aktív módszer célja a rejtett üzenetek csatornából történ½o kivonása, kicserélése, üzenetek tartalmának módosítása (üzenetmódosítás), vagy legális rendszerelemtől információk kicsalása (megszemélyesítés).

Átrendezés, behelyettesítés, Caesar módszer

A titkosítás úgy történik, hogy az üzenet minden betűjét k betűvel eltoljuk (mod 26), illetve azzal helyettesítjük. A dekódoláskor a betűket k pozícióval "visszatoljuk".

Általános behelyettesítés – kulcs bevezetése

Szimetrikus kulcsú titkosítás:

- Az üzenet küldője ugyanazzal a kulccsal titkosítja az üzenetet, mint amivel az üzenet fogadója visszafejti azt.

- A legnagyobb probléma az ilyen típusú titkosító módszerekkel, hogy mivel ugyanarra a kulcsra van szükség mind a két oldalon, ezért titkosítás előtt valamilyen biztonságos csatornán (például: személyesen) meg kell egyezni a kulcsot illetően.
- Elméletileg a kulcsot csak az üzenet küldője és a fogadója ismeri, ezért csak ők tudják visszafejteni a nyílt üzenetet.
- Ebből az is következik, hogy az üzenetet csak az küldhette, aki ismeri a titkosításhoz használt kulcsot.

Előnyök:

- Gyorsabb, nagy méretű adat átvitelére alkalmasabb.
- A kisebb méretű kulcsoknak (56-256 bit) köszönhetően kisebb tárhelyet igényel.

Hátrányok:

- A kulcs mind a két oldalon azonos, ezért azt előzetesen meg kell osztani biztonságos csatornán keresztül, valamint minden két résztvevő közötti kommunikáció esetén eltérő kulcsra van szükség.
- Könnyebben feltörhetőek a kisebb méretű kulcsok miatt, gyakoribb kulcsváltoztatásra van szükség a biztonság érdekében.
- Nincs lehetőség digitális aláírásra.

Szimmetrikus kulcsú titkosítási algoritmusok:

- DES (Data Encryption Algorithm)
- AES

Aszimmetrikus kulcsú titkosítás:

- Aszimmetrikus kulcsú titkosítás esetén a legfontosabb eltérés a szimmetrikus kulcsú titkosításhoz képest, hogy minden kommunikációban résztvevő személynek két kulcsa van, ezeket nyilvános kulcsnak és titkos kulcsnak nevezzük.
- Amíg a szimmetrikus módszerek esetén mindenki ugyanazzal a kulccsal végez titkosítást és visszafejtést az üzeneteken, aszimmetrikus titkosításnál megfelelő működés esetén mind a két kulcsnak eltérőnek kell lennie két felhasználó esetén.
- A nyilvános kulcs, mint a neve is mutatja bárki számára hozzáférhető, központi helyen tárolódik. A titkos kulcsot csak az a személy birtokolhatja, aki a tulajdonosa. Kiemelten fontos, hogy sose kerüljön senki más kezébe.

Előnyök:

- Minden résztvevő egyetlen kulcspárral rendelkezik, amellyel az összes résztvevővel tud kommunikálni.
- Nehezebben törhetőek fel a nagyobb méretű kulcsok miatt, nincs szükség gyakori kulcsváltoztatásra.
- Lehetőség van digitális aláírásra.

Hátrányok:

- Lassabb, kisebb méretű, de kritikusabb adat átvitelére alkalmasabb.
- A nagyobb méretű kulcsok (1024-2048 bit) miatt nagyobb tárhely szükséges.

PL: RSA (Ron Rivest, Adi Shamir és Len Adleman), DSA (Digital Signature Algorithm) , ELGAMAL

Miért probléma a nyelvi statisztika?

Az emberi nyelvek az emberi nyelvek redundánsak. Nem minden betű egyformán gyakori. Az angolban az E,T,A,O…. a leggyakrabbak. A magyarban pedig az E, A és a T. Igen ritka az Ő, W, X és a Q.

Ezek a statisztikák a nyelvekre jellemzőek, segítségükkel a nyílt szöveg nyelve azonosítható. Vagy gyakori/jellemző szavak keresése. Kriptoanalízis folyamán a gyakran ismétlődő szavak segítségével visszafejthető az üzenet.

Miért probléma az egyszeri kulcsos titkosítás?

- A kulcs mind a két oldalon azonos, ezért azt előzetesen meg kell osztani biztonságos csatornán keresztül, valamint minden két résztvevő közötti kommunikáció esetén eltérő kulcsra van szükség.
- Könnyebben feltörhetőek a kisebb méretű kulcsok miatt, gyakoribb kulcsváltoztatásra van szükség a biztonság érdekében.

Vigenère-rejtjel

Kódolni, illetve dekódolni úgy tudunk egy szöveget, hogy először is alá írjuk folytonosan a kulcs szöveget (vagy egy számsort). Ezek után összeadjuk a két betűt úgy, mintha a táblázat egy szorzótábla lenne. Ha a

	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
A	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z
B	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A
C	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B
D	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C
E	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D
F	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E
G	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F
H	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G
I	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H
J	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I
K	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J
L	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K
M	M	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L
N	N	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M
O	O	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N
P	P	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O
Q	Q	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P
R	R	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q
S	S	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R
T	T	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S
U	U	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T
V	V	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U
W	W	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V
X	X	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W
Y	Y	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X
Z	Z	A	B	C	D	E	F	G	H	I	J	K	L	M	N	O	P	Q	R	S	T	U	V	W	X	Y

kulcsszámot tartalmaz, akkor ki kell keresni, hogy az abc-ben melyik betű van azon a sorszámon. Kódolás esetén kiválasztjuk azt az oszlopot, aminek az első eleme az eredeti szöveg aktuális betűje, és azt a sort, aminek az első betűje az éppen a kulcs aktuális betűje. A metszetűk megadja a rejtjelet.Dekódolás esetén kiválasztjuk azt a sort, aminek az első betűje a kulcs aktuális betűje.
Majd kikeressük ebben a sorban a szöveg aktuális betűjét. Az eredeti betű az előbbi betűt tartalmazó oszlop első eleme lesz.

PL:

kulcs: kuki

Eredeti szöveg : takacs marta

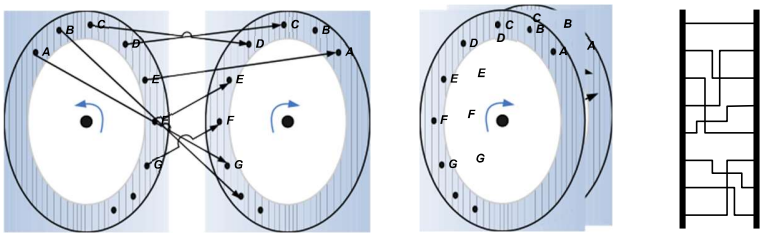
kukiku kikuk

Eredmény: duuimm wibnk

Enigma – keverő algoritmus résszel bővült

(forrás: Virrasztó Tamás – Titkosítás és adatrejtjelezés)

Az Enigma a II. világháborúban a németek által használt titkosítógép volt, működésének alapelve a rotor-elv. Egy rotort úgy kell elképzelni, mint két korongot, melyek egymáshoz vannak rögzítve, együtt forognak, rajtuk egyenként 26 elektromos érintkező van. A szemben lévő érintkezőit tetszőlegesen kötögessük össze valahogy így:

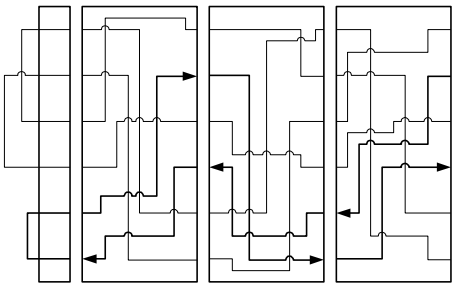


Ha egy elektromos jel a „szendvics” bal oldalán az A érintkezőre érkezik, akkor a „szendvics” másik oldalán a δ betűnél fog távozni, vagyis ez egy egyábécés helyettesítés. Most fogjunk kettő ilyen rotort, és tegyük őket egymás mellé. Az elsőn A-> δ, a másodikon δ ->μ lesz a jel útja. Majd bonyolítsuk egy kicsit a helyzetet: minden egyes jel után az egyik rotor lépjen egyet! Ha körbefordult, lépjen egyet a következő rotor is!

A gép a következő elemekből állt:

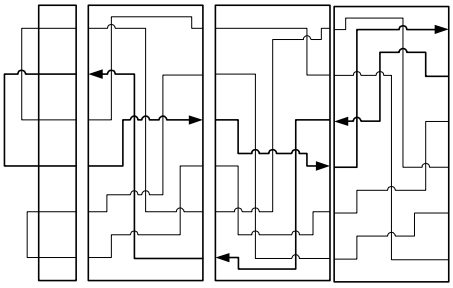
- egy 26 latin betűs billentyűzet (1)
- kapcsolótábla (Stecker) – a kezdeti keverés biztosítéka
- három forgó tárcsa (rotor) + reflektor
- egy 26 lámpás kijelző, ami a titkosítás és a megfejtés eredményét mutatta

Az elrendezési mód gyakorlatilag három helyettesítő eszköz soros összekötésének felel meg. Tételezzük fel, hogy az alábbi ábrán a billentyűzet a III. rotorhoz csatlakozik, és egy B betűt nyomtak meg (a kapcsolótábla keverő hatását most nem vesszük figyelembe). A III. rotoron a B-ből E lesz, a II. rotoron a E-ből delta, végül az I. rotoron a delta-ből sigma. Ezután az I. rotor sigma betűjét az úgynevezett „reflektoron" keresztül visszavezetjük erre a „rotorszendvicsre". Azt, hogy a kilépő sigma betű miként fog visszatérni a rotorokhoz, szintén kézzel lehetett huzalozni, vagyis a reflektor is szabadon konfigurálható volt.



Reflektor 1.rotor 2.rotor 3.rotor

Alaphelzet



Reflektor 1.rotor 2.rotor 3.rotor

A 3.rotor lépett egyet

Rotorok

– három rotor, egyenként 26 lehetséges pozícióval: 26×26×26 = ***17576***

– három rotor a következő 6 sorrendben rakható egymás mellé: ***6***

(1,2,3), (1,3,2), (2,1,3), (2,3,1), (3,1,2), (3,2,1)

Stecker - kapcsolótábla

– a 26 betűs ábécé hat betűpárja rengeteg módon cserélhető meg:

$(26\times25 \times 24\times23 \times 22\times21 \times 20\times19 \times 18\times17 \times 16\times15) / (6! \times 2^6) =$

100 391 791 500

$100391791500 \times 6 \times 17576 =$ ***10 586 916 764 424 000***

Öt tárcsa esetén, amiből tetszőleges hármat választhatunk, ez a szám pontosan tízsze-resére emelkedik...

III. MODERN MÓDSZEREK

(forrás: INFORMÁCIÓ- ÉS KÓDELMÉLET – GALÁNTAI AURÉL)

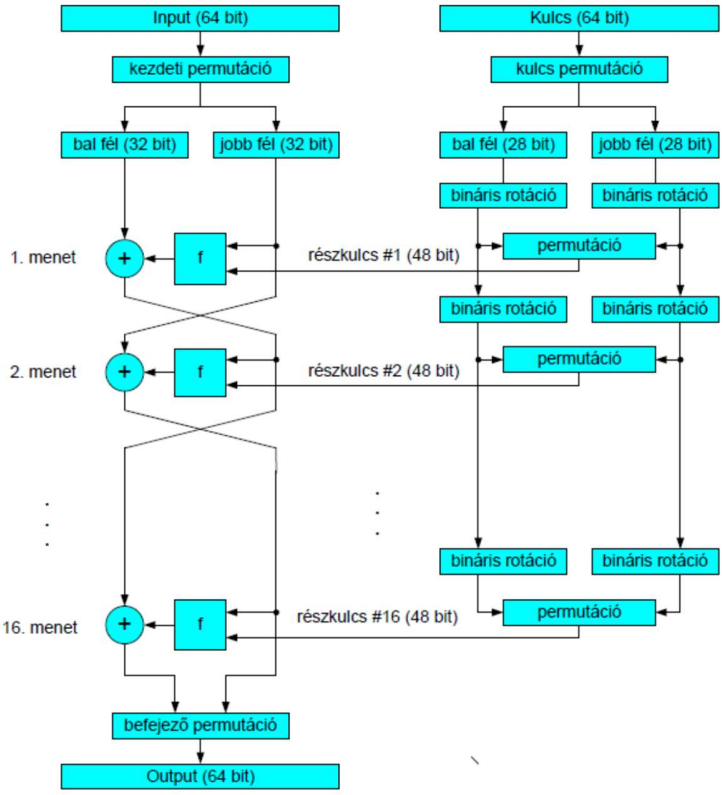
DES

A DES az egyik legelterjedtebb blokkonkénti IBM eredetű titkosító szabvány.

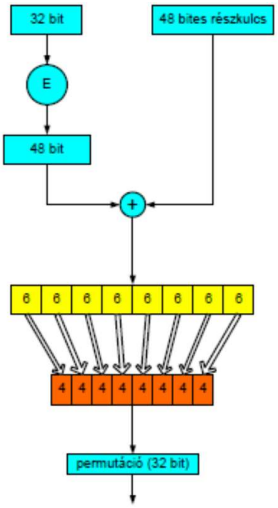
Jellemzői:

- 64-bites blokkokat kódol
- A kulcshossz: 56 bit.
- 16-szor megismételt ciklusban:
  - A szöveget 2 részre bontja
  - Az egyik fél szöveget a kulcs és az f függvény segítségével kombinálja
  - XOR kombináció a másik fél szöveggel
  - Felcseréli a két részt.

Az algoritmus:



Az ábrán az **f**-el jelölt blokk egy Feistel-függvényt jelöl, amelynek sémája:



Az **f függvény** egy fél blokkon (32) dolgozik és 4 lépést hajt végre:

1. **Kiterjesztés** (**Expansion**): A 32 bites blokkot kiterjeszti 48 bitesre, bizonyos bitek kettőzésével.

2. **Kulcskeverés** (**Keymixing**): A kapott eredményt az XOR művelet felhasználásával keveri egy részkulccsal. 16 részkulcsot használ. Minden egyes menetben egyet. A részkulcsok előállítását (iterációját) az első DES ábra tartalmazza.
3. **Helyettesítés** (**Substitution**): A keverés után a blokkot 8 darab 6 bites részre bontja, amelyek mindegyikét az S-dobozok (substitution-box) 4 bites kimenetté transzformálnak (nem-lineáris transzformáció lookup táblázat formájában). Ezek adják a DES biztonsági magját. Enélkül a DES lineáris lenne és triviálisan megfejthető.
4. **Permutáció**: Az S-dobozokból kijövő 32 kimenetet egy rögzített permutációval permutálja.

Gyengeségei:

1. Komplementálási tulajdonság: Ek (x) = y , Ek (x) = y, ahol x az x bitenkénti komplemense.
2. Négy gyenge kulcsa van
3. Hat félig gyenge (k1; k2) kulcspárja van
4. Támadások:
    a. Brute force támadás: minden lehetséges kulcs végig próbálása. Speciális hardverekkel
    b. többször is feltörték (1998-ban 56 óra, 1999-ben 22 óra 15 perc, 2007-ben COPACOBANA 6.4 nap).
    c. Gyorsabb támadások:
        i. Differenciál kriptoanalízis
        ii. Lineáris kriptoanalízis
        iii. Javított Davis támadás

Kulcsmegosztás problémája (Nyivános- és magánkulcsok)

Ugyan az mint a fenti kulcsos téma.

Nyilvános kulcsok módszere

A nyílvános kulcsú titkosítás célja, hogy olyan partnerek közti kommunikáció esetén is gyakorlati titkosságot nyújtson, akik az üzenetküldés el½ott semmiféle titkos kulcsot nem cseréltek egymással. A felhasználók ugyanazt az E titkosító és D dekódoló algoritmust használják. A k_{pA} nyilvános kulcsot egy nyílvános, mindenki által hozzáférhető adattárban tároljuk.

Lépései:

- B felhasználó kiolvassa A nyilvános kulcsát a nyilvános kulcstárból.
- B elküldi a c = E (m; kp
- kriptogrammot A-nak (egy nyilvános csatornán).
- A a saját titkos kulcsát felhasználva dekódolja az üzenetet: m = D(c; ksA).

Követelményei:

- Adott m és kpA esetén legyen könny½u a c = E (m; kpA) kiszámítása.
- Ne legyen könny½u az üzenet megfejtése, ha csak a c kriptogram ismert.
- Adott c kriptogram és ksA esetén legyen könny½u m = D(c; ksA) kiszámítása.

RSA eljárás és módszere (http://slideplayer.hu/slide/2107814/)

Az RSA-eljárás nyílt kulcsú (vagyis „aszimmetrikus”) titkosító algoritmus. Ez napjaink egyik leggyakrabban használt titkosítási eljárása. Az eljárás elméleti alapjait a moduláris számelmélet és a prímszámelmélet egyes tételei jelentik.

Működése:

Az RSA-titkosításhoz egy nyílt és egy titkos kulcs tartozik. A nyílt kulcs mindenki számára ismert, s ennek segítségével kódolhatják mások nekünk szánt üzeneteiket. A nyílt kulccsal kódolt üzenetet csak a titkos kulccsal tudjuk „megfejteni”. Az RSA-eljáráshoz a következő módon generáljuk a kulcsokat:

1. Véletlenszerűen válasszunk két nagy prímet, p-t és q-t
2. Számoljuk ki N = p*q-t.
    a. **N** lesz a modulusa mind a nyilvános, mind a titkos kulcsnak is.
3. Számoljuk ki az Euler-féle φ függvény értékét N-re: φ (N) = (p-1)(q-1)
4. Válasszunk egy olyan egész számot, e-t melyre teljesül 1 < e < φ (N) és e és φ (N) legnagyobb közös osztója **1**. Azaz LNKO[1](e, φ (N)) = 1.
    a. Az **e**-t nyilvánosságra hozzuk, mint a nyilvános kulcs kitevője.
5. Számítsuk ki d-t, hogy következő kongruencia teljesüljön, $d * e \equiv 1\ (mod\ \varphi\ (\mathrm{N}))$. Azaz d*e = 1 + k * φ (N) valamely k pozitív egészre.
    a. **d** -t titokban tartjuk mint a titkos kulcs kitevője.
6. A **nyilvános kulcs** az **N** modulusból és a nyilvános **e** kitevőből áll.
7. A **titkos kulcs** az **N** modulusból és a titkos **d** kitevőből áll, melyeket természetesen nem osztunk meg mással.
8. A hatékonyság érdekében a **titkos kulcs** egy más formáját szokás tárolni:
    a. **p** és a **q**: a kulcskészítéshez szükséges prímek,
    b. **d** mod(p-1) és **d** mod(q-1)-et gyakran dmp1 és dmq1-nek nevezik.
    c. $q^{-1}\ mod(p)$-pedig iqmp-nek
9. A titkos kulcs minden része ezek alapján titokban tartandó. **p** és **q** nagyon fontosak, hiszen ezek a faktorai **N**-nek, és **d** gyors kiszámolását teszik lehetővé adott **e** által (mely ugyebár nyilvános).
10. Az N szám bináris alakban írt bitjeinek a száma adja a rejtjelzőkód hosszúságát, ami a gyakorlatban általában n=512, 1024, 2048 szokott lenni.
11. Fontos megjegyezni, hogy **e** és φ közötti relatív prím kapcsolat azért fontos, mert ez biztosítja, hogy a **d * e = 1 +k * φ(N)** diofantoszi egyenletnek van megoldása, s az könnyen meg is kapható az euklideszi algoritmus segítségével!
12. Népszerű választás **e**-re a $2^{16} + 1 = 65537$. Néhány alkalmazásnál ehelyett **e**-t 3, 5, vagy 35-nek választják inkább. Ez azért hasznos mert kisebb eszközökön meggyorsíthatja a számolást, például bankkártyákon. Viszont ezek a kisebb nyilvános kitevők nagyobb kockázati tényezőket okoznak.

[1] LNKO = Legnagyobb Közös Ösztó

Digitális aláírás

Két alapvető cél:

- Az üzenet feladójának igazolása (hitelesítés) (Védekezés aktív támadó ellen)
- Az üzenet-tartalom változatlanságának igazolása (integritásvédelem)

Az aláírás tartalma:

Az aláírás tartalmaz egy ellenőrző összeget, amihez szükség van egy MD algoritmusra (hash függvény) ez általában SHA-1 vagy MD5. Ez a dokumentumból egy fix hosszúságú [bit]sorozatot készít. Ehhez hozzáfűzzük az aláíró nevét vagy azonosítóját, az aláírás idejét, az MD algoritmus nevét, és amit még fontosnak tartunk.

A létrehozás, ellenőrzés:

Ezután ezt az aláíró kódolja a titkos kulcsával. Az így előállt kód csak a küldő titkos kulcsával állítható elő és csak az ő nyilvános kulcsával olvasható el. Ezt a kódot csatolnunk kell az üzenethez és így kell elküldeni.

Ha címzett vagy akárki, aki meg akar bizonyosodni az üzenet hitelességéről, annak kell készíteni az üzenetről egy ellenőrző összeget, ugyanazzal az MD algoritmussal, amit az aláíráskor is használtak, és dekódolnia kell az aláírást a küldő nyilvános kulcsával. Az ebben található kódot össze kell hasonlítani az újjal.

Ha megegyeznek, akkor biztos lehet abban, hogy a küldő írta alá az üzenetet, és az nem változott meg mióta alá lett írva, feltételezve, hogy nem került illetéktelen kezekbe a titkos kulcs.

IV. A TITKOSÍTÁS ALAP FOGALMAI

Titkosítás
Titkosítási modell és alapfogalmak

Az elküldeni kívánt eredeti szöveg a nyílt szöveg (plaintext). Ebből a titkosítási eljárás létrehozza a titkosított szöveget (ciphertext) vagy más néven kriptogramot . A titkosítási eljárás paramétere a kulcs (key). Ennek ismerete nélkül a megfejtés nem lehetséges (vagy legalábbis nehéz). A ciphertext-et egy nyílt csatornán továbbítjuk, ahol a betolakodó hozzáférhet ahhoz. A vevő a megfejtési módszert alkalmazza a ciphertext-re, a kulcs-csal paraméterezve. A titkosítási kulcs és a megfejtési kulcs nem feltétlenül azonos, de ebben az esetben összetartoznak. A megfejtési eljárás eredményeképpen az eredeti nyílt szöveg áll elő.

A betolakodónak (intruder) két fajtája ismert: az egyik csak hallgat, és tudomást kíván szerezni az eredeti szövegről (passzív), a másik meg akarja az üzenetet hamisítani, vagy hamis üzeneteket próbál előállítani a feladó nevében (aktív). A fenti megkülönböztetés szerint szokás a lehallgatás, illetve a megszemélyesítés elleni védelemről beszélni. A jó titkosításnak mindkét próbálkozást ki kell védenie. Az üzenet, vagy a módszerek és kulcs(ok) megfejtésére irányuló eljárás a kriptoanalízis.

A fenti eljárásokkal foglalkozó tudomány a titkosítástan vagy kriptológia.

Titkosítási igények

A titkosítás lehetőségeit egyaránt igénybe vették katonák, országukkal vagy szövetségeseikkel ésrintkező diplomaták, szemérmes naplóírók és titkolózó szerelmesek.

A téma arculatát a katonák határozták meg. A titkosítást hagyományosan rosszul fizetett hivatalnokok végezték, ezért csak egyszerű algoritmusok voltak alkalmazhatók. Újabban: Számítógépek alkalmazása, intelligens programok, ezért az algoritmusok bonyolultabbak, a törekvés a könnyen megjegyezhető kulcsokra (pl. jelszó) helyeződött át.

Igazi biztonságot csak az a titkosítás nyújt, amely tetszőleges mennyiségű választott nyílt szöveg esetén is feltörhetetlen.

Klasszikus rejtjelezések

Egyábécés rejtjelezések

a->d, b->e, stb.
A rejtjelzés a karthágóiakat becsapta, de azóta se senkit.

Caesar-féle rejtjelezés

A példákban a rövid ékezetes magyar ábécét alkalmaztam, ennek megfelelően a Caesar-rejtjelzés táblázata az alábbi:

```
plaintext: aábcdeéfghijklmnoöpqrstuüvwxyz
ciphertext:          deéfghijklmnoöpqrstuüvwxyzaábc
```

Az egyszerűség kedvéért az írásjelek és a szóközök változatlanul mennek át.

k-eltolás

Általánosított formája: az ábécét nem három, hanem tetszőleges *k* betűvel eltolva alkalmazzuk. Itt *k* a rejtjelezés kulcsa. Legfeljebb 30 féle kulcs lehet (rövid magyar ábécé), tehát a szöveget próbálkozással sem nehéz megfejteni.

általános egyábécés rejtjelezés

Tovább általánosítva: A kulcs az a táblázat, hogy melyik szimbólumot melyikre cseréljük. Összesen $4*10^{26}$ féle kulcs lehet. Az összes eset végigpróbálása: 1µs / kulcs esetén is 10^{13} év lenne.

A megfejtés valódi módszere: nyelvi statisztika (betűk, betűpárok és szavak gyakorisága). A módszer hátránya: a kulcs nehezen megjegyezhető.

Vigenére-féle rejtjelezés	Ötlet: Az egymást követő betűkre különböző egyábécés kulcsot alkalmazzunk Előnye: könnyen megjegyezhető kulcsszó. Hátrány: a kulcs hosszának megsejtésekor a probléma visszavezethető az egyábécés titkosításra és elegendő titkos szöveg rendelkezésre állásakor statisztikai módszerrel ismét elemezhető. Számítógéppel mindez nem problémát. Áthidalható ez a probléma, ha a kulcs hosszabb, mint a szöveg, de ennek a gyakorlati alkalmazása erősen korlátozott, azonkívül le kell írni, mert nem megjegyezhető. Betűk helyett nagyobb egységek kódolása is lehetséges, például betűpároké vagy szavaké.
Felcseréléses rejtjelezés	Gyakorisági statisztika elárulja a tisztán felcseréléses rejtjelezést. A kulcs a felcserélés algoritmusa.
Számítógépes titkosítási módszerek	A titkosítással kapcsolatban éppen úgy, mint minden más területen az Interneten két jól elkülönülő irányzat figyelhető meg: a kereskedelmi és az akadémiai-kutatói szemlélet. A vállalkozások, kisebb és nagyobb cégek sok hasznos termékkel jelentkeznek, de az ő információs forrásaik rendszerint a reklámnak vannak alárendelve. Az akadémiai/kutatói szemléletre és a lelkes magánemberekre jellemző a pártatlanabb látásmód, az ingyen rendelkezésre bocsátott termékek, írásos anyagok, szellemi tőke stb. Az Internet hagyományos értékei a demokratizmus, segítőkészség, önzetlenség és nyílt kommunikáció - ha élünk ezekkel - sokat segítenek abban, hogy egy-egy konkrét esetben megfelelő megoldást találjunk. A biztonsági kérdések esetén különösen fontos például, hogy tudomást szerezzünk arról, ha a kívánt titkosítási eljárást feltörték. Az Interneten sok kereskedelmi termék titkosításának gyengeségéről szerezhetünk tudomást. Például a Word, az Excel, a Wordperfect, az Unix crypt, a PKZIP, a Microsoft Money, a Lotus 1-2-3 és még egy sor más termék titkosítása

	feltörhető. Az ehhez szükséges eszközök az Internet segítségével könnyen hozzáférhetőek. Fontos, hogy a naiv szemlélő számára a jó és a rossz titkosítás nem különböztethető meg egymástól. Ezért ellenőrizetlen titkosítási módszer alkalmazása esetén fennáll a hamis biztonság illúziójának veszélye.
Adattitkosítási szabvány	A helyettesítés és felcserélés mellett a hangsúly a bonyolultabb algoritmusokra tevődött át. USA kormányának 1977-ben elfogadott szabványa a DES. 56 bites kulcs parametrizálja, 64 bites blokkokat titkosít 64 bites blokkokká. Ez valójában egy egyábécés rejtjelezés, amely 64 bites jeleket használ. A DES nemcsak blokkos, hanem folyamatos módszerrel is alkalmazható (pl. terminálok). Előny: a titkosított szöveg a szöveg teljes történetétől függ.
A DES ellentmondásossága	Kezdettől fogva gyanítják, hogy az 56 bites kulcs túl rövid, könnyű feltörni. Az IBM eredeti tervében 128 bit szerepelt, amit a Nemzetbiztonsági Hivatal kérésére csökkentettek 56 bitre. Okát nem tették közzé. A DES tervezési elvei ugyancsak titkosak. A minap sikerült egy konkrét példát feltörni, igaz, a mai viszonyokhoz képest rendkívül nagy számítási kapacitást alkalmaztak hozzá, az interneten sok ezer számítógép között szétosztva a feladatot.
A kulcs-szétosztás problémája	Minden olyan titkosítási módszernél jelentkezik a probléma, ahol a titkosítási és a megfejtési kulcs azonos. A kulcsban az adónak és a vevőnek meg kell állapodnia, erre a célra rendszerint ugyanaz a nem biztonságos

csatorna áll rendelkezésre, mint a további kommunikációra. Ráadásul a kompromittálódott kulcsot rendszeresen újra kell cserélni.

A kulcsot titkos csatornán kell átvinni, például személyes találkozás. Ez megnehezíti az azonnali és a globális kommunikációt. Egy megoldás a kulcshierarchia alkalmazása, azaz a kulcs továbbítása egy magasabb szintű szuperkulccsal, amelynek a cseréjére ritkán kerül sor. Ez elsősorban szervezeteken belül járható út. Idegen szervezetek között megnehezíti a kommunikációt.

Kulcsvédelem

Vállalati jogosultság, darabolt információ: polinommódszer.

Nyilvános kulcsú titkosítás

Szerencsére azonban van mód a borítékolásra (titkosítás) és a digitális aláírásra (hitelesítés) is. Sőt, e két dolog szorosan összefügg, ugyanaz az eszköz alkalmas. Ez az eszköz a nyilvános kulcsú titkosítás. Ennek elvét írjuk le alább.

Alapelvek

Nyilvános kulcsú titkosításnál minden egyes felhasználóhoz két kulcs tartozik: egy titkos és egy nyilvános. A titkos és a nyilvános kulcs szerepe szimmetrikus. Ha N jelöli a nyilvános kulcs alkalmazását, T a titkos kulcsét, és x egy kódolandó információ, akkor

$N(T(x))=x$ és $T(N(x))=x$

A módszer lényege, hogy rendkívül nehéz T-ből N-et meghatározni, továbbá T nem törhető fel választott nyílt szöveggel. Ezeket a követelményeket teljesíti például az MIT-n kidolgozott RSA algoritmus.

Minden felhasználónak generálnia kell a maga részére egy nyilvános/titkos kulcs párt. Ezután a nyilvános kulcsot minél szélesebb körben ismertté kell tenni, a titkos kulcsra értelemszerűen vigyázni kell.

Bárki, aki titkosított üzenetet akar küldeni, nem kell mást tegyen, mint a fogadó nyilvános kulcsával kódolnia kell az üzenetet. A nyilvános kulcs

ismerete nem segít abban, hogy a titkos kulcsot megfejtsük, ezért ha egy üzenetet valaki nyilvános kulcsával kódoltunk, akkor már magunk sem tudjuk azt visszafejteni, csakis a fogadó.

Ha hitelesíteni akarunk egy üzenetet, akkor pedig a saját titkos kulcsunkat használjuk. Az üzenetből képezünk egy az üzenetnél jóval rövidebb számot, amit az üzenet ellenőrző összegének, "ujjlenyomatának" is nevezhetünk. Ezt a számot kódoljuk azután a saját titkos kulcsunkkal. A fogadó ezt csakis a mi nyilvános kulcsunkkal tudja "kinyitni" és így biztos lehet abban, hogy az üzenetet valóban mi küldtük. Az üzenet ilyen esetben nincs feltétlenül kódolva, de mivel az egész üzenet ujjlenyomatát tartalmazza az aláírásunk, az üzeneten végrehajtott minden változtatás, egyetlen vesszőcske beszúrása vagy elhagyása is kiderül a fogadó oldalon. Ilyen módon - hasonlóan ahhoz, mint amikor aláírunk valamit - a hitelesítéssel nem csak azt garantálhatjuk, hogy kitől származik az üzenet, hanem azt is, hogy az pontosan ugyan az. Ez alkalomadtán - akár csak az aláírás - arra is alkalmas, hogy valamit a fejünkre olvassanak, mint általunk elismert, vállalt dolgot. Következésképpen a titkos kulcsunkra nem csak azért kell vigyáznunk, hogy a nekünk küldött üzeneteket ne fejtsék meg illetéktelenek, hanem azért is, hogy mások ne tudjanak "okirathamisítást" végrehajtani a kárunkra.

RSA algoritmus

A szerzők nevének kezdőbetűiből: Rivest-Shamir-Adleman “public key criptography”

Biztonság alapelve: nagy számok tényezőkre bontásának nehézsége. Példa: 200 jegyű szám felbontása a mai számítógépekkel 4 milliárd évig tart.

Válasszunk két nagy prímszámot, amelyek > 10^{100} n = p*q és z = (p-1)*(q-1) legyen d z-hez képest relatív prím keressünk egy olyen e-t, amelyre e*d mod z = 1 A titkos kulcs a (d, n) pár, a nyilvános kulcs az (e, n) pár Kódolás $C = P^e \bmod n$ Dekódolás: $P = C^d \bmod n$

A kódolás fix méretű blokkokra tördelve történik, a kódolandó blokkok $\log_2$ n-nél kevesebb bites egységek.

A kód feltöréséhez n-et fel kellene bontani p-re és q-ra hogy z és ebből d meghatározható legyen.

Hogyan győződhetünk meg arról, hogy egy nyilvános kulcs érvényes e?

Igen fontos, hogy ha valakinek a nyilvános kulcsát használjuk, akkor biztosak legyünk abban, hogy nem hamis, lejárt, vagy érvénytelen a kulcs. Ha rossz kulcsot használunk, akkor nekünk küldött hamisított üzenetet hitelesnek hihetünk vagy illetéktelenek is olvashatják titkosnak szánt üzenetünket A legegyszerűbb, és legbiztonságosabb, ha személyesen cserélünk kulcsot. Megfelelő megoldás, ha valaki a névjegyén közli, na nem a nyilvános kulcsát, de annak egy ujjlenyomatát. Szokás elektronikus levélben, vagy a Finger szolgáltatás segítségével közölni nyilvános kulcsot.

Mint bármi más információt, nyilvános kulcsokat is alá lehet digitálisan írni, akár több embernek is. Ez az ötletet kétféleképpen is ki szokták használni.

Bizalmi háló

Ha olyan aláírással kapunk egy nyilvános kulcsot, amit hitelesnek tekintünk, akkor magát ezt a nyilvános kulcsot is elfogadhatjuk. Az ilyen bemutatott kulcs aztán újabb kulcsokat hitelesíthet. Ez a “bizalmi háló” a PGP nevű népszerű, szabadon terjeszthető titkosítási programcsomagra jellemző. Ennél azt is szabályozhatjuk, hogy milyen mélységben fogadunk el “bemutatott által bemutatottakat”, és hogy hány hitelesnek ismert bemutató bemutatása kell ahhoz, hogy egy nyilvános kulcsot hitelesnek ismerjünk el.

Kulcs aláírási összejövetel

Kulcs hitelesítés egyik módja a kulcs aláírási összejövetel (key signing party). Egy ilyen összejövetelen nem kell a kulcsok tényleges aláírásának megtörténnie. A lényeg, hogy meggyőződjünk arról, mely kulcsok mely személyekhez tartoznak. A kulcsokat ilyenkor az ujjlenyomatok (fingerprint) segítségével, az esetleg ismeretlen személyeket pedig valamilyen igazolványuk (személyi, útlevél, jogosítvány) alapján azonosítjuk. Az összejövetel végén a résztvevők magukkal visznek egy-

egy papírt, amin jelölik, hogy mely személyek és ujjlenyomatok összetartozásáról győződtek meg. Otthon azután letölthetik egy kulcsszerverről azokat a kulcsokat, amiket alá szándékoznak írni, (megkaphatják a nyilvános kulcsokat más módon, pl. levelezési lista segítségével is). Ha az ujjlenyomat egyezik, akkor a kulcsot aláírva visszaküldhetik a kulcsszervernek (vagy a kulcs tulajdonosnak, vagy a levelezési listára).

Az aláírási összejövetelen egy lehetséges eljárás a következő: a résztvevőknek sokszorosítva kiosztják az aláírásra jelölt kulcsok ujjlenyomatát. Mindenki elhozza saját kulcsának ujjlenyomatát magával. Akinek szerepel a kiosztott papíron a kulcsa, az feláll, és elmondja, hogy a kiosztott papíron valóban az ő kulcsának ujjlenyomata áll-e. Személyazonosságát valamilyen igazolvánnyal igazolja (ismerősök esetén ez elmaradhat). A résztvevők ezután a kiosztott papíron bejelölik a kulcsát, mint aláírásra érdemest.

Kulcs szerverek, kulcshitelesítő autoritás

Ha egy közjegyző vagy valamilyen hivatal hitelesíti digitális aláírásával valakinek a nyilvános kulcsát, akkor ezt nem csak a polgári életben, hanem az államigazgatásban és a jogban is használhatjuk. Több ilyen szervezet, cég van már, amelyik nyilvános kulcsok hitelesítésével foglalkozik. Ilyenek a Verisgn, a Four11 vagy Európában a Deutsche Telekomm egy leányvállalata. A hálózaton ezeken kívül is számos kulcs szerver érhető el, ahonnan személyek, szerverek, intézmények nyilvános kulcsát tölthetjük le. A PGP kulcs szerverek a világon elszórtan, de szinkronban működnek.

A kétkulcsos titkosítás helyzete a világban

Sürgető igény

Manapság az Interneten legtöbbször még mindig olyan módon zajlik a kommunikáció, ahogy a “közönséges" életben elfogadhatatlannak tartanánk: nincs garancia arra, hogy onnan jön az információ, ahonnan hisszük. Ha valóban onnan is jön, nincs garancia arra, hogy - akár

rosszindulatúan - nem olvasták-e el, részben vagy egészben nem változtatták-e meg illetéktelenek.

Technikailag viszonylag egyszerű és régen ismert ezeknek a problémáknak a megoldása. Ez a nyilvános kulcsú titkosítás, mely a digitálisan tárolt, illetve továbbított információk titkosságát, hitelességét, épségét, letagadhatatlanságát
egyaránt garantálja.